

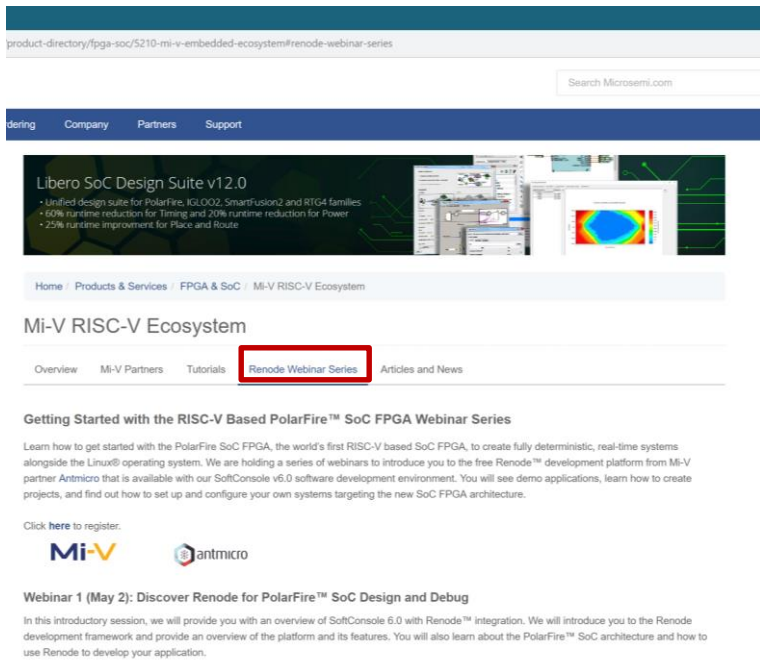
# First / Second Thursdays

---

- Nov. 7 - Webinar 7: How to Write Custom Models – Filters, Offloading, Acceleration, etc.**
- Dec. 5 - Webinar 8: Refresh SC 6.2**
- Jan. 9 - Webinar 9: Bare metal?**
- Feb. 13 - Webinar 10: Build Applications for Linux on PolarFire SoC**
- Mar. 12 - Webinar 11: Introduction to PolarFire SoC MSS Configuration and Software Flow**
- Apr. 9 - Webinar 12: Two Baremetal Applications on PolarFire SoC**
- May. 14 - Webinar 13: Linux + Real-Time (AMP Mode) on PolarFire SoC**

# Supporting Content

[www.microsemi.com/Mi-V](http://www.microsemi.com/Mi-V) “Renode Webinar Series”



The screenshot shows the Microsemi website's navigation and content. The top navigation bar includes links for Ordering, Company, Partners, and Support. Below this, a banner for the Libero SoC Design Suite v12.0 is displayed. The main content area is titled "Mi-V RISC-V Ecosystem" and features a navigation menu with links for Overview, Mi-V Partners, Tutorials, Renode Webinar Series (highlighted with a red box), and Articles and News. Below the menu, the section "Getting Started with the RISC-V Based PolarFire™ SoC FPGA Webinar Series" is visible, followed by a paragraph of introductory text and a link to register. At the bottom, the "Webinar 1 (May 2): Discover Renode for PolarFire™ SoC Design and Debug" is listed, with a brief description of the session.

product-directory/fpga-soc/5210-mi-v-embedded-ecosystem/renode-webinar-series

Search Microsemi.com

Ordering Company Partners Support

Libero SoC Design Suite v12.0

- Unified design suite for PolarFire, IGLOO2, SmartFusion2 and RTG4 families
- 60% runtime reduction for Timing and 20% runtime reduction for Power
- 25% runtime improvement for Place and Route

Home Products & Services FPGA & SoC Mi-V RISC-V Ecosystem


### Mi-V RISC-V Ecosystem

Overview Mi-V Partners Tutorials **Renode Webinar Series** Articles and News

#### Getting Started with the RISC-V Based PolarFire™ SoC FPGA Webinar Series

Learn how to get started with the PolarFire SoC FPGA, the world's first RISC-V based SoC FPGA, to create fully deterministic, real-time systems alongside the Linux® operating system. We are holding a series of webinars to introduce you to the free Renode™ development platform from Mi-V partner Antmicro that is available with our SoftConsole v6.0 software development environment. You will see demo applications, learn how to create projects, and find out how to set up and configure your own systems targeting the new SoC FPGA architecture.

Click [here](#) to register.

**Mi-V** 

#### Webinar 1 (May 2): Discover Renode for PolarFire™ SoC Design and Debug

In this introductory session, we will provide you with an overview of SoftConsole 6.0 with Renode™ integration. We will introduce you to the Renode development framework and provide an overview of the platform and its features. You will also learn about the PolarFire™ SoC architecture and how to use Renode to develop your application.

Webinar 1: Discover Renode for PolarFire® SoC Design and Debug

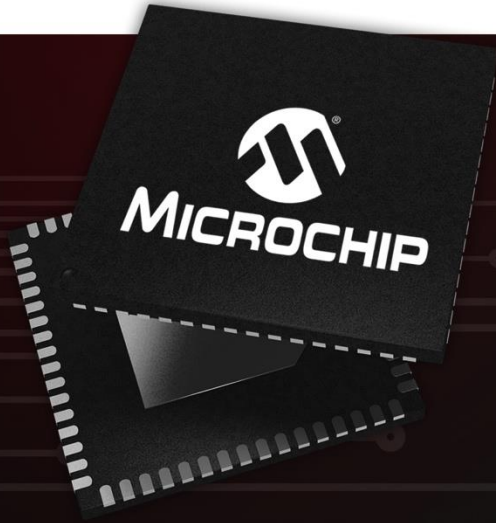
Webinar 2: How to Get Started with Renode for PolarFire SoC

Webinar 3: Learn to Debug a Bare-Metal PolarFire SoC Application with Renode

Webinar 4: Tips and Tricks for Even Easier PolarFire SoC Debug with Renode

Webinar 5: Add and Debug PolarFire SoC models with Renode

Webinar 6: Add and Debug a Pre-Existing model in PolarFire SoC



A Leading Provider of Microcontroller, Security,  
Mixed-Signal, Analog & Flash-IP Solutions



**Getting Started with the RISC-V Based PolarFire® SoC FPGA Webinar Series**  
**Session 7: “How to Write Custom Models – Filters, Offloading, Acceleration, etc”**

*Hugh Breslin, Embedded Linux Engineer*

*Thursday Nov. 7, 2019*

# Agenda

---

- **Recap: Ways to add Models**
- **Model Template**
- **Creating and Testing a Basic Register**



# Ways to Add Models



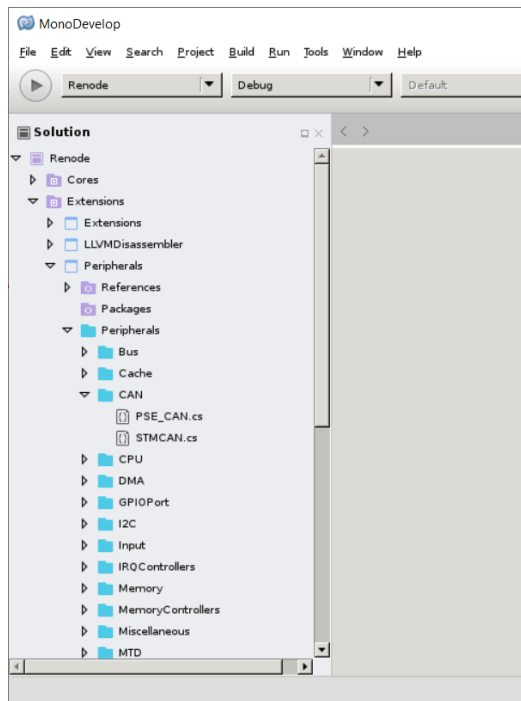
# Ways to Add Models

---

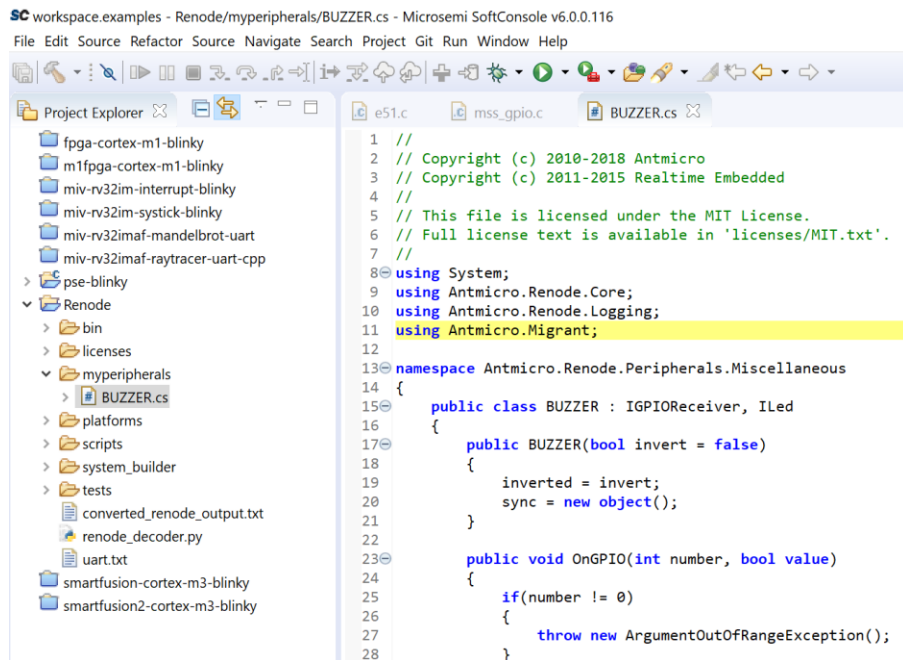
Pre-Compiled	JiT (Just-in-Time compiled)
Built in with Renode	Standalone file separate to Renode
Run faster	Can be modified without having to re-build, just relaunch
Can't be edited without rebuilding Renode	Run slower than pre-compiled
Models included with Renode are precompiled	Develop models using JiT and then build them into Renode
Available on Windows® and Linux	Only available on Linux

# Ways to Add Models

## Pre-Compiled



## JiT



# How to Add a Just-in-Time (JIT) Compiled Model

---

- **Include the C# file for the model**
  - include @[path\_to\_file]
- **Add the model to the system**
  - machine LoadPlatformDescriptionFromString  
“[sysbus\_name]: [namespace].[name] @ sysbus [address]”

```
polarfire.repl ❷  
53 mmuart0: UART.NS16550 @ sysbus 0x20000000  
54   wideRegisters: true  
55   IRQ -> plic@90  
56
```

```
[sysbus_name]: [namespace].[name] @ sysbus [address]
```



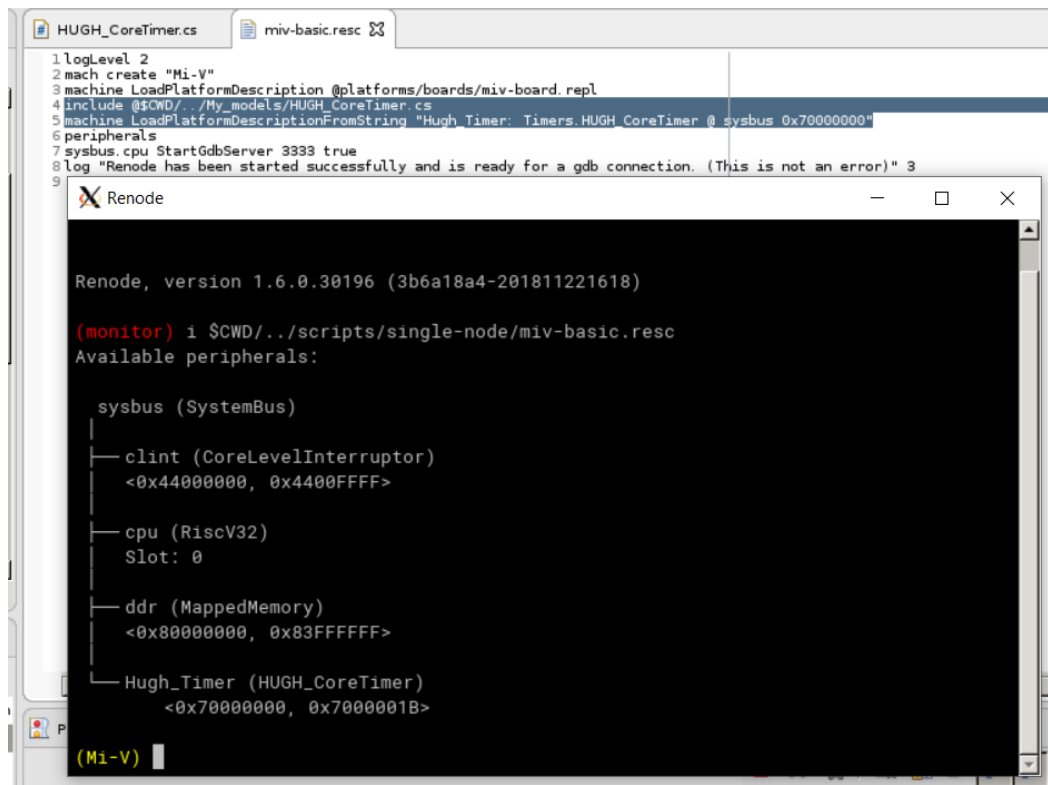
# How to Add a Just-in-Time (JIT) Compiled Model

---

- **Example commands for the Renode console:**

1. `include $CWD/../../My_models/HUGH_CoreTimer.cs`
2. `machine LoadPlatformDescriptionFromString  
"Hugh_timer: Timers.Hugh_CoreTimer @sysbus  
0x70000000"`

# How to Add a Just-in-Time (JIT) Compiled Model



```
1 logLevel 2
2 mach create "Mi-V"
3 machine LoadPlatformDescription @platforms/boards/miv-board.repl
4 include @$CWD/./My_models/HUGH_CoreTimer.cs
5 machine LoadPlatformDescriptionFromString "Hugh Timer: Timers.HUGH_CoreTimer @sysbus 0x70000000"
6 peripherals
7 sysbus.cpu StartGdbServer 3333 true
8 log "Renode has been started successfully and is ready for a gdb connection. (This is not an error)" 3
9
```

```
Renode, version 1.6.0.30196 (3b6a18a4-201811221618)
(monitor) i @$CWD/./scripts/single-node/miv-basic.resc
Available peripherals:

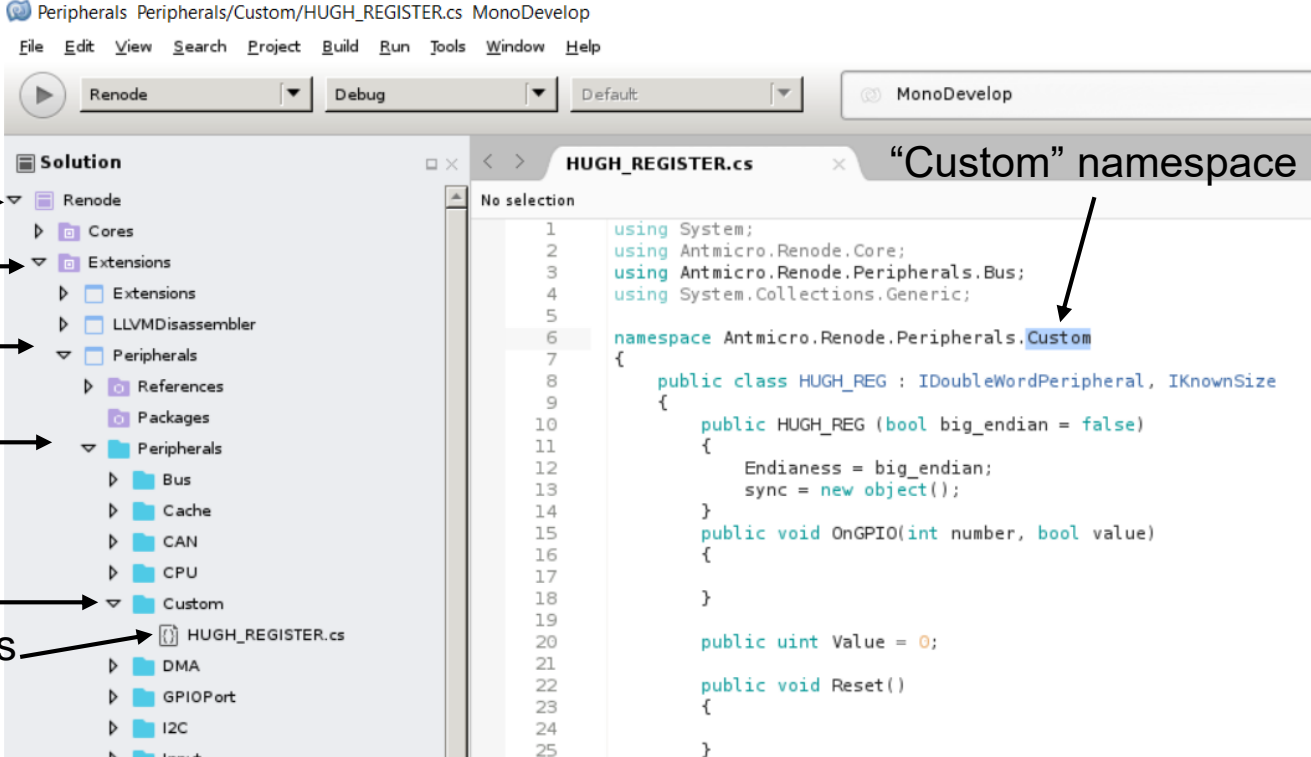
  sysbus (SystemBus)
  |
  |--- clint (CoreLevelInterruptor)
  |    <0x44000000, 0x4400FFFF>
  |
  |--- cpu (RiscV32)
  |      Slot: 0
  |
  |--- ddr (MappedMemory)
  |      <0x80000000, 0x83FFFFFF>
  |
  |--- Hugh_Timer (HUGH_CoreTimer)
  |      <0x70000000, 0x7000001B>
  |
(Mi-V) |
```

**Alternatively add commands  
to the launch script**

```
include @$CWD/./My_models/HUGH_CoreTimer.cs
```

```
machine LoadPlatformDescriptionFromString  
"Hugh_timer: Timers.Hugh_CoreTimer @sysbus  
0x70000000"
```

# How to Compile a Model



The screenshot shows the MonoDevelop IDE interface. On the left is the 'Solution' explorer, and on the right is the code editor for 'HUGH\_REGISTER.cs'.

**Solution Explorer (Left):**

- Renode solution (indicated by an arrow)
- Extensions solution (indicated by an arrow)
- Peripherals project (indicated by an arrow)
- Peripherals folder (indicated by an arrow)
- 'Custom' folder (indicated by an arrow)
- Models (indicated by an arrow, pointing to HUGH\_REGISTER.cs)

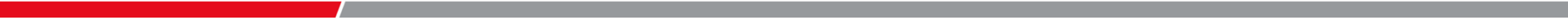
**Code Editor (Right):**

The code file is 'HUGH\_REGISTER.cs'. The namespace is 'Antmicro.Renode.Peripherals.Custom', which is highlighted by an arrow and labeled '“Custom” namespace'. The code defines a class 'HUGH\_REG' that implements 'IDoubleWordPeripheral' and 'IKnownSize'.

```
1 using System;
2 using Antmicro.Renode.Core;
3 using Antmicro.Renode.Peripherals.Bus;
4 using System.Collections.Generic;
5
6 namespace Antmicro.Renode.Peripherals.Custom
7 {
8     public class HUGH_REG : IDoubleWordPeripheral, IKnownSize
9     {
10         public HUGH_REG (bool big_endian = false)
11         {
12             Endianess = big_endian;
13             sync = new object();
14         }
15         public void OnGPIO(int number, bool value)
16         {
17         }
18     }
19
20     public uint Value = 0;
21
22     public void Reset()
23     {
24     }
25 }
```



# Model Template



# Model Template

```
using Antmicro.Renode.Peripherals.Bus;

namespace Antmicro.Renode.Peripherals /* NAMESPACE DECLARATION */
{
    public class /* CLASS DECLARATION */ : /* INTERFACE DECLARATIONS [ 1, 2, 3, ... ] */
    {
        /* VARIABLE DECLARATIONS */

        public void Reset()
        {
            /* ACTION ON RESET */
        }

        /* INTERFACE FUNCTIONS */

        /* PROPERTY DECLARATIONS */
    }
}
```

# Model Template

```
using Antmicro.Renode.Peripherals.Bus;
```

```
namespace Antmicro.Renode.Peripherals./* NAMESPACE DECLARATION */
```

```
{  
    public class /* CLASS DECLARATION */ : /* INTERFACES [ 1, 2, 3, ... ] */  
    {
```

```
        /* VARIABLE DECLARATIONS */
```

```
        public void Reset()  
        {
```

```
            /* ACTION ON RESET */
```

```
        }
```

```
        /* INTERFACE FUNCTIONS */
```

```
        /* PROPERTY DECLARATIONS */
```

```
    }  
}
```

```
using Antmicro.Renode.Peripherals.Bus;
```

```
namespace Antmicro.Renode.Peripherals.Custom
```

```
{  
    public class BASIC_REGISTER : IDoubleWordPeripheral, IKnownSize  
    {
```

```
        private uint Value = 0;
```

```
        public void Reset()  
        {
```

```
            Value = 0;
```

```
        }
```

```
        public void WriteDoubleWord(long offset, uint value)
```

```
        {
```

```
            Value = value;
```

```
        }
```

```
        public uint ReadDoubleWord(long offset)
```

```
        {
```

```
            return Value;
```

```
        }
```

```
        public long Size => 0x1C;
```

```
    }
```



# **Creating and Testing a Basic Register**



# Creating and Testing a Basic Register

---

- **Basic Register Model**
- **Adding the Model**
  - Interfacing with Sysbus
  - Testing
  - Adding Properties
  - Adding Parameters





# Basic Register Model

---

```
using Antmicro.Renode.Peripherals.Bus;
```

```
namespace Antmicro.Renode.Peripherals.Custom // Custom namespace
```

```
{
```

```
    public class BASIC_REGISTER : IDoubleWordPeripheral, IKnownSize // Declare class and interface
```

```
    {
```

```
        private uint Value = 0; // Declare variable
```

```
        public void Reset()
```

```
        {
```

```
            Value = 0; // Define action on reset
```

```
        }
```

```
        public void WriteDoubleWord(long offset, uint value) // Define method from IDoubleWordPeripheral
```

```
        {
```

```
            Value = value; // Value will take on the value passed to this function
```

```
        }
```

```
        public uint ReadDoubleWord(long offset) // Define method from IDoubleWordPeripheral
```

```
        {
```

```
            return Value; // Return current value
```

```
        }
```

```
        public long Size => 32; // Size is address space on sysbus from IKnownSize
```

```
    }
```

```
}
```



# **Adding the Model: Interfacing with Sysbus**



# Adding the Model: Interfacing with Sysbus

Declare class : interface0, interface 1.....

public class BASIC\_REGISTER : IDoubleWordPeripheral, IKnownSize

IDoubleWordPeripheral:

I: Interface

DoubleWord: 32 bit transactions

IKnownSize:

I: Interface

KnownSize: Address space

```
namespace Antmicro.Renode.Peripherals.Bus
{
    public interface IDoubleWordPeripheral : IBusPeripheral
    {
        uint ReadDoubleWord(long offset);
        void WriteDoubleWord(long offset, uint value);
    }
}
```

```
using Antmicro.Renode.Peripherals.Bus;

namespace Antmicro.Renode.Peripherals
{
    public interface IKnownSize : IBusPeripheral
    {
        long Size { get; }
    }
}
```



# Adding the Model: Interfacing with Sysbus

```
using Antmicro.Renode.Peripherals.Bus;
```

```
namespace Antmicro.Renode.Peripherals.Custom // Custom namespace
```

```
{  
    public class BASIC_REGISTER : IDoubleWordPeripheral, IKnownSize // Declare class and interface
```

```
{  
    private uint Value = 0; // Declare variable
```

```
    public void Reset()
```

```
{  
        Value = 0; // Define action on reset  
    }
```

```
    public uint ReadDoubleWord(long offset) // Define method from IDoubleWordPeripheral
```

```
{  
        return Value; // Return current value  
    }
```

```
    public void WriteDoubleWord(long offset, uint value) // Define method from IDoubleWordPeripheral
```

```
{  
        Value = value; // Value will take on the value passed to this function  
    }
```

```
    public long Size => 32; // Size is address space on sysbus
```

```
}  
}
```

```
namespace Antmicro.Renode.Peripherals.Bus
```

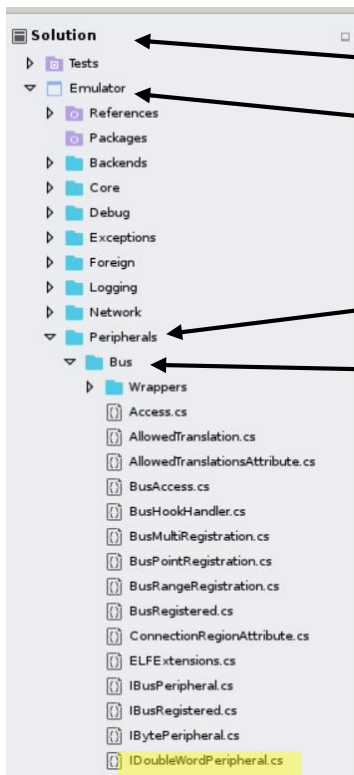
```
{  
    public interface IDoubleWordPeripheral : IBusPeripheral  
    {  
        uint ReadDoubleWord(long offset);  
        void WriteDoubleWord(long offset, uint value);  
    }  
}
```

```
using Antmicro.Renode.Peripherals.Bus;
```

```
namespace Antmicro.Renode.Peripherals
```

```
{  
    public interface IKnownSize : IBusPeripheral  
    {  
        long Size { get; }  
    }  
}
```

# Adding the Model: Interfacing with Sysbus



Renode solution

Emulator solution

Peripherals folder

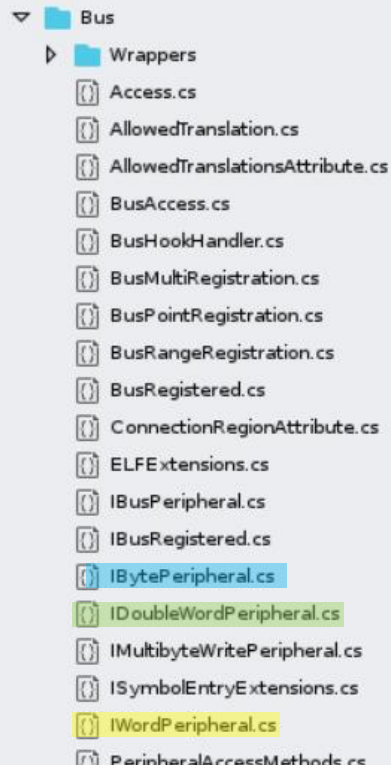
Bus folder

```
using Antmicro.Renode.Peripherals.Bus;

namespace Antmicro.Renode.Peripherals
{
    public interface IKnownSize : IBusPeripheral
    {
        long Size { get; }
    }
}
```

```
namespace Antmicro.Renode.Peripherals.Bus
{
    public interface IDoubleWordPeripheral : IBusPeripheral
    {
        uint ReadDoubleWord(long offset);
        void WriteDoubleWord(long offset, uint value);
    }
}
```

# Adding the Model: Interfacing with Sysbus



```

1 //
2 // Copyright (c) 2010-2018 Antmicro
3 // Copyright (c) 2011-2015 Realtime Embedded
4 //
5 // This file is licensed under the MIT License.
6 // Full license text is available in 'licenses/MIT.txt'.
7 //
8
9 namespace Antmicro.Renode.Peripherals.Bus
10 {
11     public interface IBytePeripheral : IBusPeripheral
12     {
13         byte ReadByte(long offset);
14         void WriteByte(long offset, byte value);
15     }
16 }

```

```

1 //
2 // Copyright (c) 2010-2018 Antmicro
3 // Copyright (c) 2011-2015 Realtime Embedded
4 //
5 // This file is licensed under the MIT License.
6 // Full license text is available in 'licenses/MIT.txt'.
7 //
8
9 namespace Antmicro.Renode.Peripherals.Bus
10 {
11     public interface IWordPeripheral : IBusPeripheral
12     {
13         ushort ReadWord(long offset);
14         void WriteWord(long offset, ushort value);
15     }
16 }

```

```

1 //
2 // Copyright (c) 2010-2018 Antmicro
3 // Copyright (c) 2011-2015 Realtime Embedded
4 //
5 // This file is licensed under the MIT License.
6 // Full license text is available in 'licenses/MIT.txt'.
7 //
8
9 namespace Antmicro.Renode.Peripherals.Bus
10 {
11     public interface IDoubleWordPeripheral : IBusPeripheral
12     {
13         uint ReadDoubleWord(long offset);
14         void WriteDoubleWord(long offset, uint value);
15     }
16 }

```



# Adding the Model: Interfacing with Sysbus

```
using Antmicro.Renode.Peripherals.Bus;

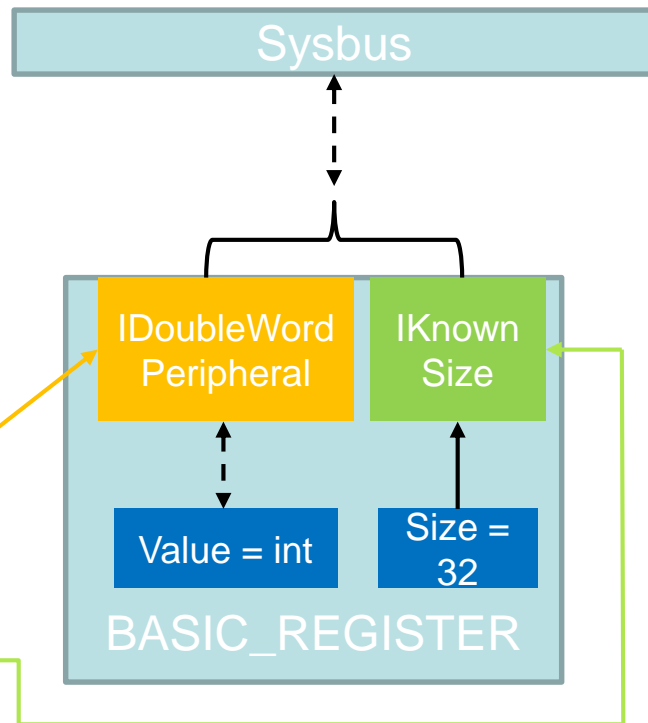
namespace Antmicro.Renode.Peripherals.Custom // Custom namespace
{
    public class BASIC_REGISTER : IDoubleWordPeripheral, IKnownSize // Declare class and interface
    {
        private uint Value = 0; // Declare variable

        public void Reset()
        {
            Value = 0; // Define action on reset
        }

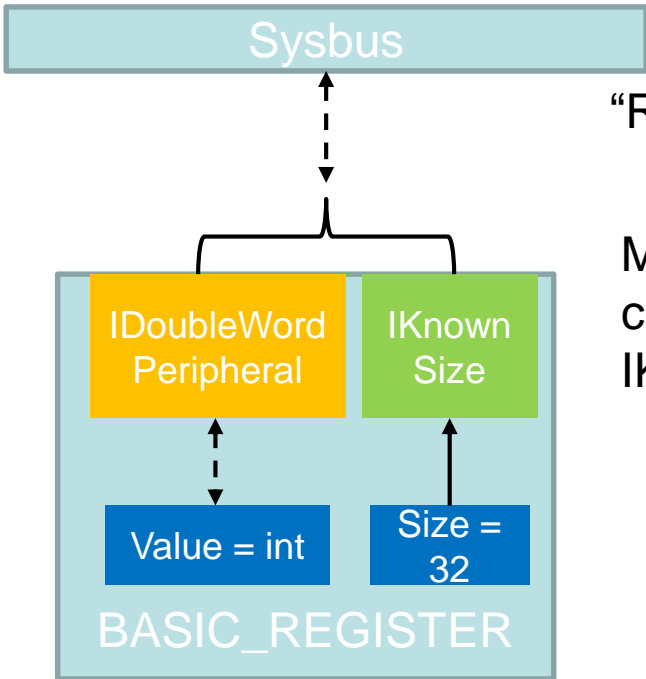
        public uint ReadDoubleWord(long offset) // Define method from IDoubleWordPeripheral
        {
            return Value; // Return current value
        }

        public void WriteDoubleWord(long offset, uint value) // Define method from IDoubleWordPeripheral
        {
            Value = value; // Value will take on the value passed to this function
        }

        public long Size => 32; // Size is address space on sysbus
    }
}
```



# Adding the Model: Interfacing with Sysbus



Machine LoadPlatformDescriptionFromString  
 “REG: Custom.BASIC\_REGISTER @ sysbus 0x70009000”

Must provide “Size” when  
 connecting to sysbus using  
 IKnownSize

Start address  
 End address?

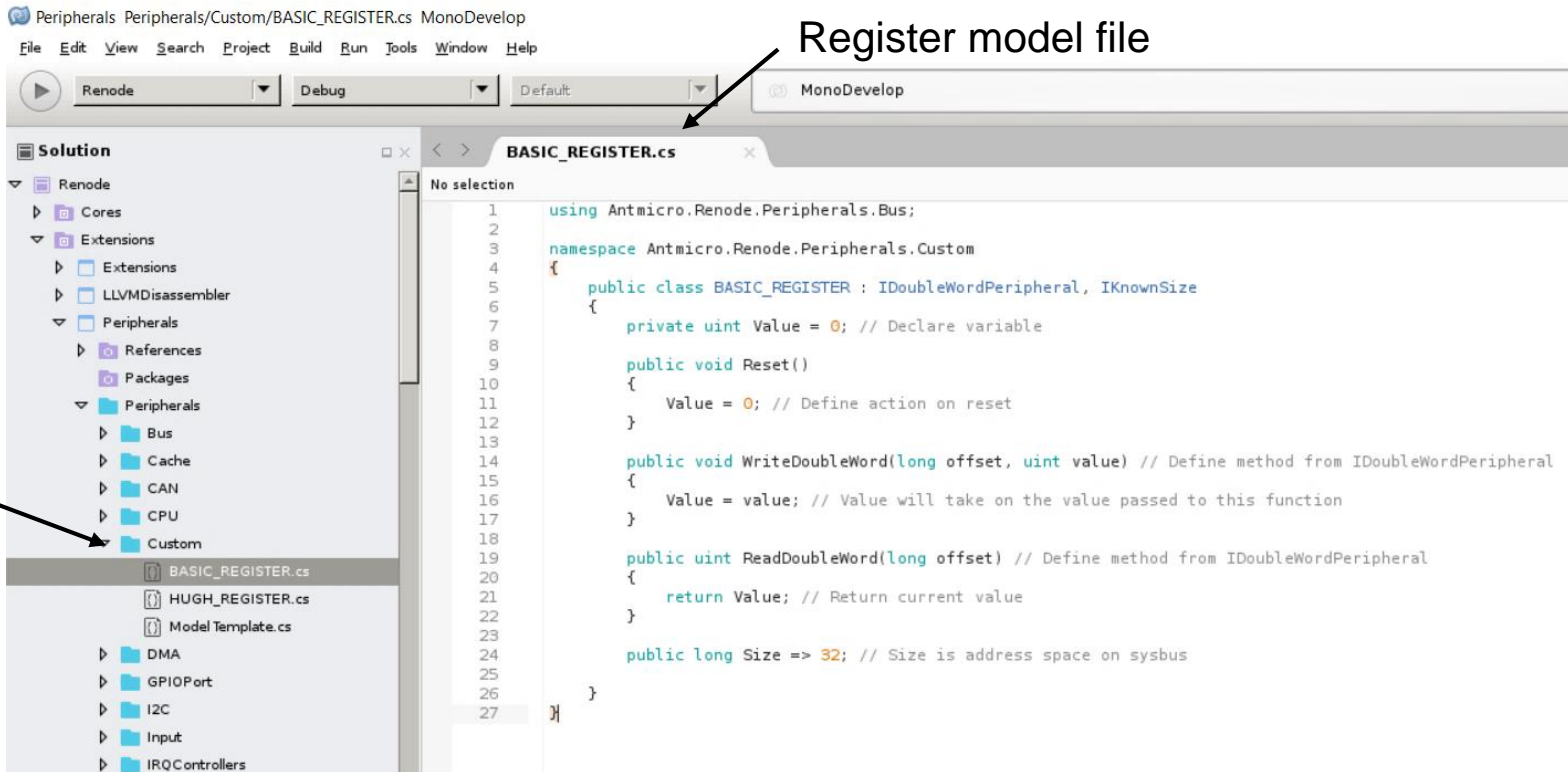
End address = start address + Size<sub>(16)</sub>

End address = 0x70009000 + 0x20<sub>(16)</sub>

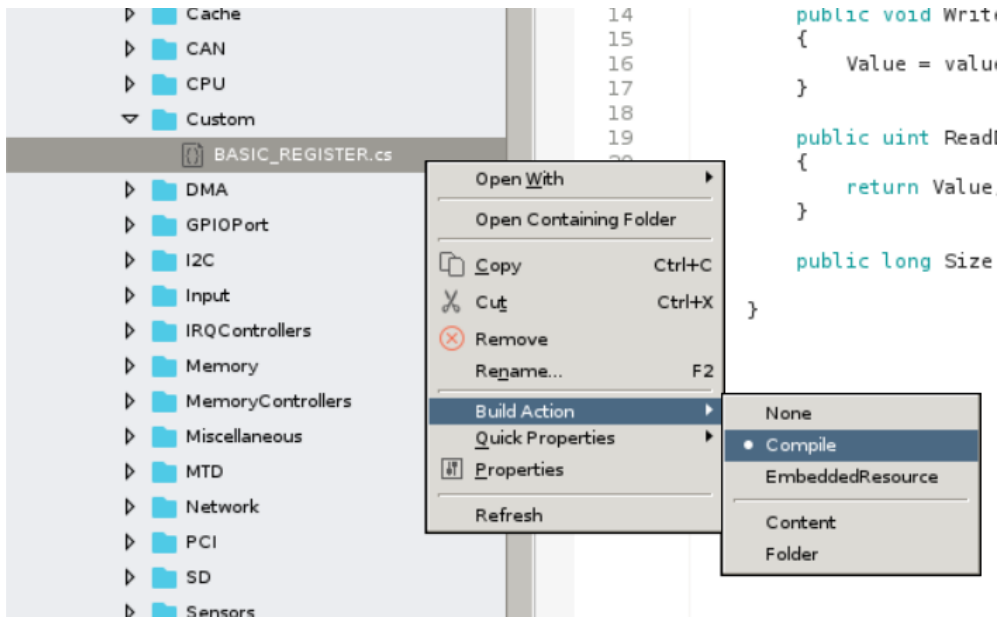
0x70009001F = 0x70009000 + 0x20<sub>(16)</sub>



# Adding the Model: Interfacing with Sysbus



# Adding the Model: Interfacing with Sysbus



Include the model in the build to be compiled

## Solution

- Renode
  - Cores
  - Extensions
    - Extensions
    - LLVMDisassembler
  - Peripherals
    - References
    - Packages
    - Peripherals
      - Bus
      - Cache
      - CAN
      - CPU
      - Custom
        - BASIC\_REGISTER.cs
      - DMA
      - GPIOPort
      - I2C
      - Input
      - IRQControllers
      - Memory
      - MemoryControllers
      - Miscellaneous
      - MTD
      - Network
      - PCI
      - SD
      - Sensors
      - SPI
      - Timers
      - UART
      - USB
      - USBDeprecated

## BASIC\_REGISTER.cs

```
1 using Antmicro.Renode.Peripherals.Bus;
2
3 namespace Antmicro.Renode.Peripherals.Custom
4 {
5     public class BASIC_REGISTER : IDoubleWordPeripheral, IKnownSize
6     {
7         private uint Value = 0; // Declare variable
8
9         public void Reset()
10         {
11             Value = 0; // Define action on reset
12         }
13
14         public void WriteDoubleWord(long offset, uint value) // Define method from IDoubleWordPeripheral
15         {
16             Value = value; // Value will take on the value passed to this function
17         }
18
19         public uint ReadDoubleWord(long offset) // Define method from IDoubleWordPeripheral
20         {
21             return Value; // Return current value
22         }
23
24         public long Size => 32; // Size is address space on sysbus
25     }
26 }
27
```

SC workspace.examples - REGISTER\_TEST/main.S - Microsemi SoftConsole v6.0.0.116

File Edit Source Refactor Navigate Search Project Run Window Help

Custom Models

- fpga-cortex-m1-bl
- m1fpga-cortex-m
- miv-rv32im-interru
- miv-rv32im-systicl
- miv-rv32imaf-mar
- miv-rv32imaf-rayt
- pse-blinky
- REGISTER\_TEST
  - Binaries
  - Includes
  - Debug
  - drivers
  - hal
  - riscv\_hal
  - src
  - hw\_platform.h
  - main.S
- smartfusion-corte
- smartfusion2-corte

main.S

```
1.global main
2
3main:
4
5    li x5, 0x70009000
6
7    li x7, 0xff00ff00
8    li x8, 0x00ff00ff
9    li x9, 0xffff0000
10
11    lw x10, (x5)
12
13    sw x7, (x5)
14    lw x11, (x5)
15
16    sw x8, (x5)
17    lw x12, (x5)
18
19    sw x9, (x5)
20    lw x13, (x5)
21
22    ebreak
23
24
```

Registers

Name	Value
------	-------

Problems Tasks Console Properties Debug

<terminated>REGISTER\_TEST Debug [GDB OpenOCD Debugging]

<terminated, exit value: 0>riscv64-unknown-elf-gdb

main (myppurememory)

```
<0x60000000, 0x6003FFFF>
```

gpioInputs (MiV\_CoreGPIO)

```
<0x70002000, 0x700020A3>
```

user\_switch\_0 (Button)

```
Address: 0
```

user\_switch\_1 (Button)

```
Address: 1
```

user\_switch\_2 (Button)

```
Address: 2
```

gpioOutputs (MiV\_CoreGPIO)

```
<0x70005000, 0x700050A3>
```

led0 (LED)

```
Address: 0
```

led1 (LED)

```
Address: 1
```

plic (PlatformLevelInterruptController)

```
<0x40000000, 0x43FFFFFF>
```

REG (BASIC\_REGISTER)

```
<0x70009000, 0x7000901F>
```

smallRom (MappedMemory)

```
<0x00001000, 0x0000FFFF>
```

timer0 (MiV\_CoreTimer)

```
<0x70003000, 0x7000301B>
```

timer1 (MiV\_CoreTimer)

```
<0x70004000, 0x7000401B>
```

uart (MiV\_CoreUART)

```
<0x70001000, 0x70001017>
```

(Mi-V) machine StartGdbServer 3333 true

(Mi-V) []



# **Adding the Model: Adding Properties**





# Adding the Model: Adding Properties

```
using Antmicro.Renode.Peripherals.Bus;
```

```
namespace Antmicro.Renode.Peripherals.Custom
```

```
{  
    public class BASIC_REGISTER : IDoubleWordPeripheral, IKnownSize  
    {  
        private uint Value = 0;  
  
        public void Reset()  
        {  
            Value = 0;  
        }  
  
        public uint ReadDoubleWord(long offset)  
        {  
            return Value;  
        }  
  
        public void WriteDoubleWord(long offset, uint value)  
        {  
            Value = value; // Value will take on the value passed to this function  
        }  
  
        public long Size => 32; // Size is address space on sysbus  
    }  
}
```

## IKnownSize

```
using Antmicro.Renode.Peripherals.Bus;  
  
namespace Antmicro.Renode.Peripherals  
{  
    public interface IKnownSize : IBusPeripheral  
    {  
        long Size { get; }  
    }  
}
```

```
- Void WriteDoubleWord (Int64 offset, UInt32 value)  
- Void WriteDoubleWordNotTranslated (Int64 address, UInt32 value)  
- Void WriteWordNotTranslated (Int64 address, UInt16 value)  
- Void WriteWordUsingDword (Int64 address, UInt16 value)  
- Void WriteWordUsingDwordBigEndian (Int64 address, UInt16 value)
```

Usage:

```
sysbus.REG MethodName param1 param2 ...
```

The following properties are available:

```
- Int64 Size  
    available for "get"
```

Usage:

```
- get: sysbus.REG PropertyName  
- set: sysbus.REG PropertyName Value
```

(Mi-V)

# Adding the Model: Adding Properties

public long Property1 { get; private set; } = 1;

public uint Property2 { get; set; } = 2;

public ushort Property3 { private get; set; } = 3;

public long Size => 32;

The following properties are available:

- Int64 Property1  
available for 'get'
- UInt32 Property2  
available for 'get' and 'set'
- UInt16 Property3  
available for 'set'
- Int64 Size  
available for 'get'

Usage:

- get: sysbus.REG PropertyName
- set: sysbus.REG PropertyName Value

(M1-V) REG Property1 ← Property1 get

0x0000000000000001

(M1-V) REG Property2 ← Property2 get

0x00000002

(M1-V) REG Property2 0x5 ← Property2 set 0x5

(M1-V) REG Property2 ← Property2 get

0x00000005

(M1-V) REG Property3 ← Property3 get

There was an error executing command 'REG Property3'

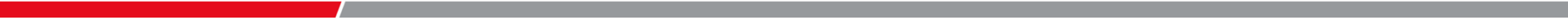
Could not execute this action on property Property3

(M1-V) REG Property3 0x3 ← Property3 set

(M1-V) █



# **Adding the Model: Adding Parameters**





# Adding the Model: Adding Parameters

```
using Antmicro.Renode.Peripherals.Bus;

namespace Antmicro.Renode.Peripherals /* NAMESPACE DECLARATION */
{
    public class /* CLASS DECLARATION */ : /* INTERFACE DECLARATIONS [ 1, 2, 3, ... ] */
    {
        public /* CLASS */ ( /* PARAMETERS */)
        {
            /* ASSIGNMENTS / ACTIONS */
        }

        /* VARIABLE DECLARATIONS */

        public void Reset()
        {
            /* ACTION ON RESET */
        }

        /* INTERFACE FUNCTIONS */

        /* PROPERTY DECLARATIONS */
    }
}
```

# Adding the Model: Adding Parameters

```
using Antmicro.Renode.Peripherals.Bus;

namespace Antmicro.Renode.Peripherals /* NAMESPACE DECLARATION */
{
    public class /* CLASS DECLARATION */ : /* INTERFACE DECLARATIONS [ 1, 2, 3, ... ] */
    {
        public /* CLASS */ ( /* PARAMETERS */)
        {
            /* ASSIGNMENTS / ACTIONS */
        }

        /* VARIABLE DECLARATIONS */

        public void Reset()
        {
            /* ACTION ON RESET */
        }

        /* INTERFACE FUNCTIONS */

        /* PROPERTY DECLARATIONS */
    }
}
```

```
using Antmicro.Renode.Peripherals.Bus;

namespace Antmicro.Renode.Peripherals.Custom
{
    public class BASIC_REGISTER_PROPERTY : IDoubleWordPeripheral, IKnownSize
    {
        public BASIC_REGISTER_PROPERTY(bool zero_reset = True)
        {
            Reset_to_zero = zero_reset;
        }

        private uint Value = 0;
        public bool Reset_to_zero;

        public void Reset()
        {
            Value = Reset_to_zero ? 0 : 1;
        }

        public void WriteDoubleWord(long offset, uint value)
        {
            Value = value;
        }

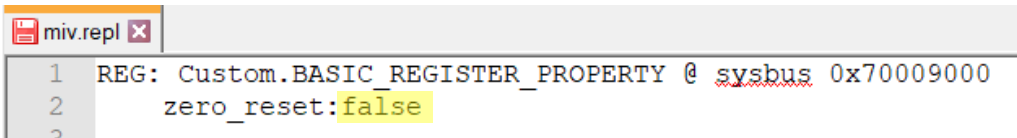
        .....
    }
}
```

# Adding the Model: Adding Parameters

```
public BASIC_REGISTER_PROPERTY(bool zero_reset = true)
{
    Reset_to_zero = zero_reset;
}
```

```
public bool Reset_to_zero;
private uint Value = 0; // Declare variable
```

```
public void Reset()
{
    if (Reset_to_zero == true)
    {
        Value = 0;
    }
    else
    {
        Value = 1;
    }
}
```



miv.repl

```
1 REG: Custom.BASIC_REGISTER_PROPERTY @ sysbus 0x70009000
2   zero_reset:false
3
```



Renode

RENODE™

Renode, version 1.8.1.27101 (eb2addec-201910141500)

```
(monitor) i $CWD/../../../../scripts/single-node/miv.resc
(MI-V)
```



# Adding the Model: Adding Parameters

```
public BASIC_REGISTER_PROPERTY(bool zero_reset = true)
{
    Reset_to_zero = zero_reset;
}
```

```
public bool Reset_to_zero;
private uint Value = 0; // Declare variable
```

```
public void Reset()
{
    if (Reset_to_zero == true)
    {
        Value = 0;
    }
    else
    {
        Value = 1;
    }
}
```

miv.repl

```
1 REG: Custom.BASIC_REGISTER_PROPERTY @ sysbus 0x70009000
2   zero_reset: true
```

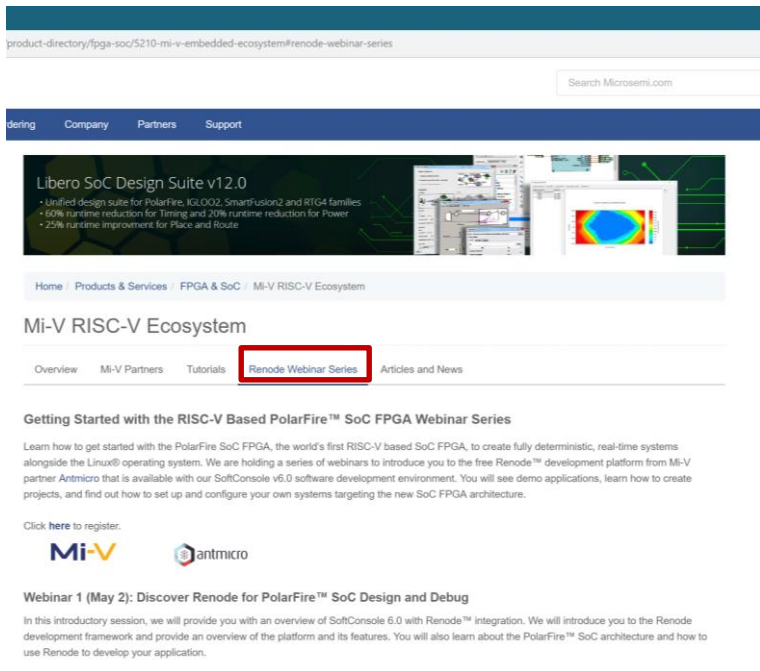


# First / Second Thursdays

---

- Dec. 5 - Webinar 8: Handling Binaries**
- Jan. 9 - Webinar 9: Run Linux® on Renode (PolarFire® SoC Model as a Quad-Core SMP) – this is not a Linux/Buildroot tutorial**
- Feb. 13 - Webinar 10: Build Applications for Linux on PolarFire SoC**
- Mar. 12 - Webinar 11: Introduction to PolarFire SoC MSS Configuration and Software Flow**
- Apr. 9 - Webinar 12: Two Baremetal Applications on PolarFire SoC**
- May. 14 - Webinar 13: Linux + Real-Time (AMP Mode) on PolarFire SoC**

## [www.microsemi.com/Mi-V](http://www.microsemi.com/Mi-V) “Renode Webinar Series”



product-directory/fpga-soc/5210-mi-v-embedded-ecosystem/renode-webinar-series

Search Microsemi.com

dering Company Partners Support

Libero SoC Design Suite v12.0

- Unified design suite for PolarFire, IGLOO2, SmartFusion2 and RTG4 families
- 60% runtime reduction for Timing and 20% runtime reduction for Power
- 25% runtime improvement for Place and Route

Home / Products & Services / FPGA & SoC / Mi-V RISC-V Ecosystem


### Mi-V RISC-V Ecosystem

Overview Mi-V Partners Tutorials **Renode Webinar Series** Articles and News

#### Getting Started with the RISC-V Based PolarFire™ SoC FPGA Webinar Series

Learn how to get started with the PolarFire SoC FPGA, the world's first RISC-V based SoC FPGA, to create fully deterministic, real-time systems alongside the Linux® operating system. We are holding a series of webinars to introduce you to the free Renode™ development platform from Mi-V partner Antmicro that is available with our SoftConsole v6.0 software development environment. You will see demo applications, learn how to create projects, and find out how to set up and configure your own systems targeting the new SoC FPGA architecture.

Click [here](#) to register.

**Mi-V** 

#### Webinar 1 (May 2): Discover Renode for PolarFire™ SoC Design and Debug

In this introductory session, we will provide you with an overview of SoftConsole 6.0 with Renode™ integration. We will introduce you to the Renode development framework and provide an overview of the platform and its features. You will also learn about the PolarFire™ SoC architecture and how to use Renode to develop your application.

Webinar 1: Discover Renode for PolarFire® SoC Design and Debug

Webinar 2: How to Get Started with Renode for PolarFire SoC

Webinar 3: Learn to Debug a Bare-Metal PolarFire SoC Application with Renode

Webinar 4: Tips and Tricks for Even Easier PolarFire SoC Debug with Renode

Webinar 5: Add and Debug PolarFire SoC models with Renode

Webinar 6: Add and Debug and Pre-Existing model in PolarFire SoC



**Thank You**

---